SAND2004-4080
Printed August 2004

Salinas Primer

Garth Reese* Manoj K. Bhardwaj† Timothy Walsh‡

Sandia National Laboratories
Albuquerque, NM 87185-0847

August 16, 2004

*Phone: (505) 845-8640
†Phone: (505) 844-3041
‡Phone: (505) 284-5374

Salinas Primer

Contents

Salinas Primer

1 Introduction

What is Salinas?

Salinas provides a massively parallel implementation of structural dynamics finite element analysis. This capability is required for high fidelity, validated models used in modal, vibration, static and shock analysis of weapons systems. General capabilities for modal, statics and transient dynamics are provided.

Salinas is similar to commercial codes like Nastran or Abaqus. It has some nonlinear capability, but excels in linear computation. It is different than the above commercial codes in that it is designed to operate efficiently in a massively parallel environment.

Why this Document?

Even for an experienced analyst, running a new finite element package can be a challenge. This little primer is intended to make part of this task easier by presenting the basic steps in a simple way. The analyst is referred to the theory manual [1] for details of the mathematics behind the work. The *User's Notes* [2] should be used for more complex inputs, and will have more details about the process (as well as many more examples). More information can be found on our web pages, 3 or 4.

Finite element analysis can be deceptive. Any software can give the wrong answers if used improperly, and occasionally even when used properly. Certainly a solid background in structural mechanics is necessary to build an adequate finite element model and interpret the results. This primer should provide a quick start in answering some of the more common questions that come up in using Salinas.

Common Questions

Issue	**section**
What is Salinas?	1, 7
Where do I get executables?	2
What inputs are required?	3
What output is produced?	4
How do I create the input?	5
How do I use the output?	5
What about running in parallel?	9
Is there an example?	10

2 Getting Help

The **Salinas** team welcomes feedback. In addition to the team members tasked with software development, we have a support individual. If at all possible, we would like to receive bug reports and other requests for help in email. To report a bug, or to request help from the **Salinas** team, send email to `salinas-help@sandia.gov`. We also maintain a user list for a broader audience, `salinas-users@sandia.gov`. You may join the user list by sending email to `majordomo@sandia.gov` with the subject "subscribe salinas-users".

Where do I get Salinas?

For those with access to Sandia internal computers, the executables are already built and available on most platforms. A list of these machines and the location of the executables is available at our web site.

`http://www.jal.sandia.gov/Salinas`

For those with no access to this web site, please contact the developers using our email expander.

`salinas-help@sandia.gov`

For smaller runs you may want to use the serial salinas package. This is named `salinas_smp_xx`, where `xx` refers to the release number.[1] Larger calculations (typically more than 100,000 degrees of freedom) are usually best done in parallel. The standard parallel executable will be named `salinas_dp_xx`.[2]

3 Files Required for Running Salinas

Salinas splits the information needed for analysis of a model into two files (or sets of files).

mesh file All the information about the mesh (nodes, element connectivity, etc) is contained in a binary **Exodus** file. In parallel this file is split into pieces so that each processor reads a file containing only the elements it will process. See section 6 for details.

The output results (displacements, stresses, etc) will be written to second **Exodus** file, which will contain the original geometry and the results.

[1] `SMP` is an abreviation for symmetric multiprocessor. Usually salinas uses only one processor on these systems.

[2] `DP` stands for FETI-DP, which is the Dual-Primal solver used for the linear solves.

text input Information about the solution process, materials, loads and boundary conditions is found in a text file (usually with a .inp extension and called an INP file). The INP file also contains a reference to the mesh file in the `geometry_file` specification. In parallel applications, each processor needs access to the information in this file.

Both the mesh file and the INP file must be available to run **Salinas** .

4 Output Data

Salinas generates two main types of output.

Text log files The operation of the code, and most output can be printed to a log file. This file has the name of the input file with the extension *.rslt*. Obviously these can become large and unmanageable. Data is also not joined by processor.

exodus results The outputs such as displacements, stresses, etc. are normally written to the output exodus file. If this is a parallel run, there is one such file written per processor. They are typically joined together before post processing (see sections 9 and 5).

5 Pre and Post processing

A variety of tools are available for pre and post processing the geometry and input files.

5.1 Preprocessing Options

- *Cubit*, a mesh generation tool from Sandia National Laboratories is perhaps the most powerful mesh generation capability for directly producing **Exodus** meshes. The *Cubit* team also distributes *verde* which can be used for mesh verification.

- *MSC/Patran* can be used with an **Exodus** preference for output and analysis of mesh files.

- *Nasgen*, available from the **Salinas** team can be used to translate a *Nastran* type mesh into both an exodusII and an INP file. Note that there are limitations with this approach, i.e. not everything translates./indexNasgen

- Some of the *seacas*[3] tools can be used to combine meshes (including equivalencing interface nodes). These tools are usually required for very large meshes, as most tools are serial and suffer severe slow down for large meshes.

Generally a text editor such as *vi* or *emacs* is used to edit and modify the text input.

5.2 PostProcessing

These options for post processing the result **Exodus** file have been used by our team. Other options are under development.

- For meshes that contain only simple 8 node hexes, *blot* may be sufficient. *Blot* is available as part of the seacas distribution at Sandia Labs. It is no longer supported, but may meet simple needs.

- Probably the most elegant post processing tool is *Ensight* from CEI. *Ensight* directly supports the **Exodus** format (along with other readers), and prepares stunningly beautiful displays. Its client/server mode and server-of-servers supports extremely large meshes.

- If you are using *MSC/Patran* for pre-processing, it can also be used as a post processing tool. It has been used successfully with meshes of a few million elements.

6 A micro-primer on ExodusII

An **Exodus** file contains a binary representation of the geometric entities in a finite element mesh. The reference material in [5] provides a complete description.

The **Exodus** database is built upon *netcdf* [6], and can be manipulated with other packages that read this format, notably *ncdump*. The database contains the following.

1. A collection of nodes (or grid points) with their coordinates. It is advantageous if the nodes are numbered 1 to N.

2. *Blocks* of element with identical topologies. Each block contains the nodal connectivity and may contain attributes for each element.

[3] SEACAS is a collection of tools for manipulation of the **Exodus** database. It is available from Sandia National Labs.

3. *Nodesets* are collections of nodes that may be used for application of loads and boundary conditions. Note that each node of the nodeset has a single multiplier (or distribution factor) that allows for spatial distribution of loads. Nodesets with distribution factors of zero cannot be used to apply loads.

4. *Sidesets* are collections of element faces and the associated nodes. The faces do not have to be topologically identical (i.e. they can contain triangles and quadrilaterals in the same set). These sidesets are used to apply loads and boundary conditions. They may also be used for contact surfaces.

5. *Results* type **Exodus** files may also contain the results of an analysis. The results are somewhat constrained in that only the time (or frequency) dimension is "unlimited". At each time value, every node has the same number of result variables. Similar constraints apply to element variables except that an element truth table determines which element blocks contain the variables.

In addition to *ncdump*, the *seacas* tool *exotxt* can be used to convert an **Exodus** file to text for examination. This translation has the advantage of clearly labelling the different quantities. The reverse operation *txtexo* works in some cases, and minor editting of the **Exodus** file is sometimes possible using these tools. For most modification, one of the preprocessing tools described above should be used.

7 How Salinas Compares with other Packages

Salinas was designed to work on the DOE/NNSA supercomputers for computation of the structural response of complex weapons components. It's strengths lie in computational scalability, a focussed effort on joint mechanics, and on its use as a platform to incorporate uncertainty quantification. The team is small. We try to be responsive to needs, but the budget is nothing like that found in commercial entities. In general, **Salinas** is a poor replacement for these products if you need solid phone support, and commercially available capabilities.

A comparison with some of the features of MSC/Nastran is shown in Table 1.

8 Running Salinas on serial UNIX platforms

On serial unix platforms, **Salinas** is run with a single argument, the ASCII input file.

```
salinas example.inp
```

The log file will be written to *example.rslt* if outputs have been specified in the ECHO section. If outputs have been specified in the OUTPUTS section, a new

Table 1: Capabilities Comparison

Salinas	**Nastran**
Massively Parallel	30+ years of verification
Scalable and Fast	Out of Core Serial solvers
Less Complete Element Libraries	Extensive Element Libraries
Small Response Team	dmap
Much smaller user base	Extensive User base

exodus file will be generated. The file name is derived from the `geometry_file` specified in the ASCII input (see section 3).

9 Running Salinas in Parallel

This section provides an example of how to perform an analysis on the Intel Teraflop (**janus**) using **Salinas**. This implies that the execution of **Salinas** will be in parallel. There is some overhead to running in parallel versus serial. Assuming a **Salinas** text input file exists and an **Exodus** file exists which contains the finite element model, the following steps are needed.

1. Decide on how many processors, *nproc*, are needed.

2. Create an input file for **nem_slice**. The partition software can be executed on a workstation to create a load balance file. The name of this file is specified in the input file for *nem_slice*, and usually has a *.nem* extension.

3. Create your workspace on **janus** on /scratch/tmp_?? - where ?? is (currently) your choice of 1 thru 10.

4. Move the **Salinas** input file, **Exodus** file, and load balance file to your work space on **janus**.

5. Create an input file for **nem_spread**. Execution of **nem_spread** (on **janus**) with this input will create *nproc* **Exodus** files from the master **Exodus** file and move them to the locations specified in the **nem_spread** input file.

6. Modify the **FILE** section of the **Salinas** input file to agree with the number of RAID disks available and the location of the subdomain **Exodus** files created

by **nem_spread**.[4]

7. Modify the **ECHO** section in the **Salinas** input file using the keyword **subdomain** to indicate which processors should produce text results files. Having all processors output text results files is very slow for large models. By default only the first subdomain will write an echo file.

8. Use the **yod** command to run **Salinas** in parallel.

9. Create an input file for **nem_join** to join your results back into one **Exodus** output file.

Each step is detailed in the following paragraphs. Additional information on parallel execution can be found at *http://jal.sandia.gov* under the **SEACAS** documentation link.

9.1 Number of Processors Needed

Running **Salinas** in parallel requires the user to specify how many processors at a minimum are needed in order to "fit" the problem into available memory on **janus**. For most applications running om compute nodes with 256MB per node, a good rule of thumb is to have approximately 4000 nodes/processor. If a finite element model has 1,000,000 nodes, use 250 processors.

9.2 Use nem_slice (or yada) to load balance the model

An example of a **nem_slice** input file is, e.g. *junk_slice.inp*,

```
Graph Type = elemental
Decomposition Method = multikl,cnctd_dom
Input ExodusII File = junk.exo
Output NemesisI File = junk.nem
#Solver Specifications
Machine Description = mesh=500
Misc Options = face_adj
#Weighting Specifications
```

[4]RAID - *Redundant Array of Inexpensive (or Independent) Disks.* These are very important to the performance of a parallel computer. Most are no longer independently addressable so the numraid should be 1.

This input file will create a load balance file, *junk.nem*, for running **Salinas** on 500 processors. Note, the *face_adj* option is useful for 3-d models to prevent mechanisms from appearing in the decomposed subdomains and is highly recommended for optimal performance.

To create the load balance file, *junk.nem*, simply type

```
workstation_prompt> nem_slice -a junk_slice.inp
```

The load balancing software, **nem_slice**, is typically executed on a serial machine such as a workstation. More detailed information on **nem_slice** is available at *http://jal.sandia.gov* under the link to the **SEACAS** documentation.[5]

9.3 Janus Work Space

To run **Salinas** in parallel, work space on **janus** is needed. On the /scratch space on **janus**, there are 10 temp directories. Simply choose one, and make a directory using your username, as follows.

```
janus> cd /scratch/tmp_1
janus> mkdir $USER
```

After the work space on janus is set up, move the **Salinas** input file, **Exodus** file, and load balance file (*junk.nem*) to it.

9.4 Using Nem_spread

The load balanced **Exodus** database must be "spread" to *nproc* mini-databases. Each processor reads from its own data file. An example **nem_spread** input file is, e.g. *junk_spread.inp*.

```
Input FEM file                  = junk.exo
LB file                         = junk.nem
Debug                           = 4
------------------------------------------------------------
                Parallel I/O section
```

[5]The **Salinas** team now distributes *yada* with its standard distribution. Yada performs the same tasks as nem_slice, but it is usually much better in keeping well connected subdomains. Run yada as follows, replacing 57 number of subdomains required.

```
workstation_prompt> yada example.exo 57
```

```
------------------------------------------------------------
Parallel Disk Info      = number=18
Parallel file location  = root=/pfs_grande/tmp_, subdir=username
```

Here, `username` must be replaced by the name of the user.

The **Exodus** file and the load balance file need to be defined in the **nem_spread** input file. There are 18 RAID disks currently available on **janus**. These are the number of disks available to which input/output can be performed in parallel. The **FILE** section in the **Salinas** input file needs to have the number of raids defined using the keyword **numraid**. Therefore, for janus, **numraid 18**, must appear in the **Salinas** input file. This number must match the parallel disk info line in the **nem_spread** input file.

If running for the first time on janus, proper directories must be established on the RAID disks. Currently, the raids are setup at */pfs_grande/tmp_??* where ?? is a number between 1 and 18 (18 raids). A few `csh` shell commands can make the required directories.

```
janus> foreach i (1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18)
foreach? mkdir /pfs_grande/tmp_$i/$USER
foreach? end
janus>
```

To execute **nem_spread**,

```
janus> /cougar/bin/yod -sz 4 nem_spread junk_spread.inp
```

This execution of **nem_spread** will spread *nproc* **Exodus** files onto the RAID disks specified in the input file for **nem_spread**. This location must also be specified in the **FILE** section of the **Salinas** input file as follows, assuming your load balance file is *junk.nem* created for 500 processors,

```
FILE
   geometry_file '/pfs_grande/tmp_%d/username/junk.par.500.%.3d'
   numraid 18
END
```

The "`%d`" after *tmp_* is used in **Salinas** in conjunction with the number of RAIDs available. The "`%.3d`" at the end of the line for the **geometry_file** is used in conjunction with how many processors the load balance file was created with. The

following table shows what must be used after *junk.par.*`nproc` for various processors requested.

Condition	Use this
nproc<10	"%.1d"
10≤*nproc*<100	"%.2d"
100≤*nproc*<1000	"%.3d"
1000≤*nproc*<10000	"%.4d"

Since **nem_spread** is a parallel code, **yod** must be used to execute it, using the -sz option to specify how many processors are needed. This number need not agree with the number of processors for execution of the analysis. Typically no more than 20 processors would be used to spread files. The **showmesh** utility can be used to indicate the number of interactive processors available.

9.5 Salinas FILE Section

If a load balance file *junk.nem* is created for execution of **Salinas** for 500 processors, and the number of raids is 18, then the **FILE** section of the **Salinas** input file must look like the following.

```
FILE
  numraid 18
  geometry_file '/pfs_grande/tmp_%d/username/junk.par.500.%.3d'
END
```

9.6 Running Salinas

Once the necessary setup has been done, and a parallel **Salinas** code exists in your work space, then

```
janus> cd /scratch/tmp_1/$USER
janus> yod -sz 500 salinas junk.inp
```

This will run **Salinas** in parallel on 500 processors using the input file *junk.inp*.[6]

In practice, only a small number of processors are available interactively on **janus**. To use a larger number of processors, the NQS queuing system must be

[6] **yod** is the command used on this platform to start a parallel run. On other platforms it may be **mpirun**(most Linux), **poe**(IBM) or some other command.

used. Help is available under the `man` pages on **janus** under the topics `qsub` and `qstat`. To submit an NQS submission, create a small shell script, such as the following.

```
janus> cat run_it
#!/bin/sh
date
cd /scratch/tmp_1/$USER
yod -sz 500 salinas junk.inp
date
```

The NQS job is submitted using `qsub` with a command such as the following.

```
/usr/bin/qsub -lT 90:00 -lP 500 -q snl.day -me run_it
```

This command submits a 90 minute run using 500 processors to the queue `snl.day`. A message will be mailed to you when the run has completed, and output from standard out and standard error will be found in files in your working directory. Status of your run can be obtained using `qstat`. Status of all NQS submissions is available with `qstat -a` or `qstat -av`. Contact *janus-help@sandia.gov* for information on queueing policies and options.

9.7 Using Nem_join

Once the analysis run has been completed, the output exodus files will need to be recombined into a single file for visualization and processing. **Nem_join** accomplishes this process. A **Nem_join** input file is very similar to the **nem_spread** input file. An example input file is, e.g. *junk_join.inp*.

```
Input FEM file                  = junk.exo
Scalar Results FEM file         = junk-out.exo
Use Scalar Mesh File            = yes
Parallel Results file base name  = junk-out.par
Number of processors            = 500
Debug                           = 4
------------------------------------------------------------
               Parallel I/O section
------------------------------------------------------------
Parallel Disk Info      = number=18
Parallel file location  = root=/pfs_grande/tmp_,subdir=username
```

To run **nem_join**, simply do the following:

```
janus> yod -sz 4 nem_join junk_join.inp
```

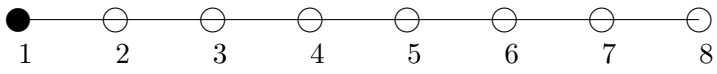
This will create a file *junk-out.exo* in your current directory by combining all the **Exodus** output files located on the RAID disks. This is a standard **Exodus** file which may be visualized and processed using serial tools.

10 Example

Sections 3 and 5 assume the existence of a preprocessor for generation of the geometry files used by Salinas. Indeed, because the exodus format is binary, it is very difficult to directly edit the geometry. However, to help illustrate the process of model construction, we will use `nasgen` to develop a model. `Nasgen` is one of the tools distributed with Salinas. It provides a translation of most of the nastran bulk data file constructs into the exodus format.

Figure 1 illustrates the model, which is composed of a simple beam with 8 grid points. Each grid is separated by 1 unit. The left end (grid 1) is clamped. We apply an axial load to grid 8.

Figure 1: Model Beam for example



A sample nastran bulk data file is shown in Figure 2. The model is translated using nasgen with the following command.

```
nasgen example.bdf example.exo -S
```

There are two files output from this run. The exodus file (which is binary) can been translated into text using exotxt. It is not shown here because even this simple model is awkward to including in this primer. A portion of the Salinas input file, `example.inp`, is shown in Figure 3. The file has been editted to retain only the necessary data.

Results are shown in Figures 4 and 5. Figure 4 contains the text of the example.rslt file including the value for the first modal frequency. Figure 5 shows the first bending mode of the beam as post processed by ensight.

References

[1] Reese, G., Bhardwaj, M., Segalman, D., Pierson, K., and Walsh, T., "Salinas – Theory Manual," Tech. rep., Sandia National Laboratories, 2004.

[2] Reese, G. and Bhardwaj, M., "Salinas – User's Notes," Tech. Rep. SAND99-2801, Sandia National Laboratories, 2003.

[3] "Salinas - Web page," http://www.jal.sandia.gov/Salinas.

[4] "Salinas - external Web page," http://www.endo.sandia.gov/9234/Salinas.

[5] Schoof, L. A. and Yarberry, V. R., "EXODUS II: A Finite Element Data Model," Tech. Rep. SAND92-2137, Sandia National Laboratories, 1994.

[6] "Netcdf," http://www.unidata.ucar.edu/packages/netcdf/.

Figure 2: Nastran Input (BDF file) for example

```
$ Direct Text Input for Bulk Data
$ Elements and Element Properties for region : beamprop
PBAR     1       1       .1      .05     .06     .11
$ Pset: "beamprop" will be imported as: "pbar.1"
CBAR     1       1       1       2       0.      1.      0.
CBAR     2       1       2       3       0.      1.      0.
CBAR     3       1       3       4       0.      1.      0.
CBAR     4       1       4       5       0.      1.      0.
CBAR     5       1       5       6       0.      1.      0.
CBAR     6       1       6       7       0.      1.      0.
CBAR     7       1       7       8       0.      1.      0.
$ Referenced Material Records
$ Material Record : al
MAT1     1       1.+7            .3      2.59-4
$ Nodes of the Entire Model
GRID     1               0.      0.      0.
GRID     2               1.      0.      0.
GRID     3               2.      0.      0.
GRID     4               3.      0.      0.
GRID     5               4.      0.      0.
GRID     6               5.      0.      0.
GRID     7               6.      0.      0.
GRID     8               7.      0.      0.
$ Displacement Constraints of Load Set : clamp
SPC1     1       123456  1
$ Nodal Forces of Load Set : f1
FORCE    1       8       0       1.      1.      0.      0.
```

Figure 3: Salinas input file, example.inp

```
SOLUTION
  eigen nmodes=1
  title 'nasgen translation from example.bdf'
END
FILE
   geometry_file 'example.exo'
END
BOUNDARY
  nodeset 11 fixed
END
LOADS
  nodeset 111
      force = 1 0 0
END
OUTPUTS
       disp
END
ECHO
//    timing
      mass
END
BLOCK 10
   material=1
   Area=0.1
   I1=0.05
   I2=0.06
   J=0.11
END
MATERIAL  1
   Isotropic
   E=1e+07
   NU=0.3
   density=0.000259
END
```

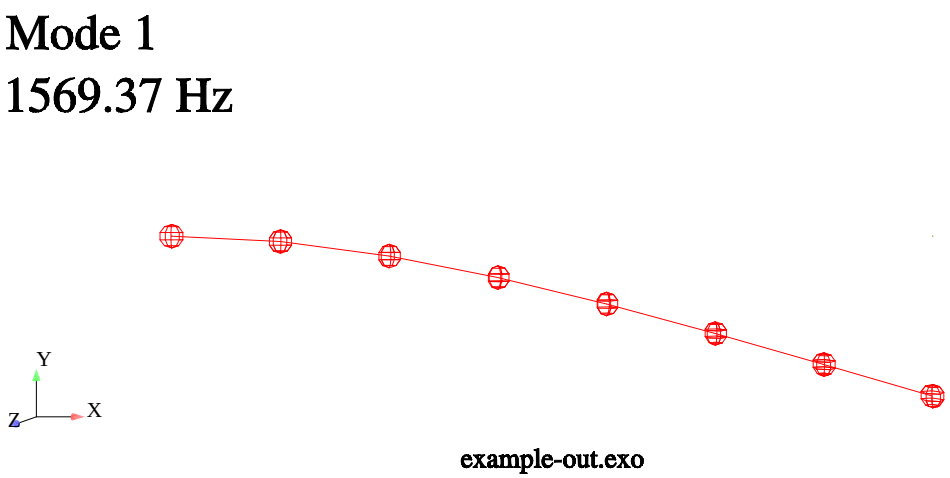
Figure 4: Example results file, example.rslt

```
Salinas  version 1.3 transition Jul 19 2004.
Build Date: Wed Aug 4 13:29:55 MDT 2004
Bld Platform/OS/compiler:  sahp5120  / Linux 2.4.20-28.9psycho  / g++
Repository date: 2004/08/03 19:46:47 $
input file: example
nasgen translation from example.bdf
we found 0 local rigid link constraint equations.


Salinas - page 2                                Wed Aug  4 15:25:23 2004

nasgen translation from example.bdf
Total Mass of Structure is 0.0001813
CG of Total Mass:
         3.5             0             0
Moment of inertia tensor:
           0             0             0
           0     0.0029915             0
           0             0     0.0029915
Proc 0 stiffness matrix diagonal max/min is 34.036364.
Global stiffness matrix diagonal max/min is 34.036364.
Eigenvalues:
1: 9.72322e+07 (1569.37)
```

Figure 5: Example plot. First Bending



Index